

- Chapitre de cours: [XSLT](#)
- Appliquer une stylesheet XSLT à un fichier XML: `xsltproc stylesheet.xml fichier.xml`

1. Appliquer la feuille de style ci-dessous à un document XML quelconque (à vous d'en créer un, minimaliste de préférence):

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    Hello World!
  </xsl:template>
</xsl:stylesheet>
```

Qu'obtient-on?

2. Modifiez le XSLT pour enlever les espaces superflus avant `Hello World!` et après le saut de ligne.
3. Modifiez le XSLT à l'aide de la balise `<xsl:output>` afin que la sortie ne contienne pas l'entête `<?xml ...>`
4. :

```
rm td9.tar.gz; wget --no-cache http://fabien.viger.free.fr/xml/td9/td9.tar.gz
tar -zxf td9.tar.gz
./1.test.sh votre.xml
```

[Corrigé](#)

1. Donner un document XML sur lequel l'application de la feuille de style XSLT suivante:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="/">
    <xsl:apply-templates select="a//b"/>
  </xsl:template>
  <xsl:template match="*">
    <xsl:value-of select="text()"/>
  </xsl:template>
</xsl:stylesheet>
```

donnera le resultat suivant (attention aux espaces et sauts de ligne!):

```
Hello
World
!
```

avec `./2.1.test.sh votre.xml`

[Corrigé](#)

2. Trouvez une source XML qui, avec ce XSLT:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```

<xsl:output method="text"/>
<xsl:strip-space elements="*" />
<xsl:template match="*">
  <xsl:choose>
    <xsl:when test="@print='true'">
      <xsl:text>Element '</xsl:text>
      <xsl:value-of select="name()" />
      <xsl:text>': </xsl:text>
      <xsl:value-of select="text()" />
      <xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>Unparsed element
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="/">
  <xsl:apply-templates select="descendant::*" />
</xsl:template>
</xsl:stylesheet>

```

... produit ce résultat:

```

Element 'pere': Hello,
Element 'fils': World
Unparsed element
Element 'fille': !

```

avec `./2.2.test.sh votre.xml`

[Corrigé](#)

- (\*) En utilisant judicieusement les règles `<xsl:apply-templates select="...">`, écrire un XSLT qui affiche le texte des enfants de la racine (uniquement: ni la racine, ni les petits-enfants); et qui les comme cela:
  - d'abord les éléments `<a>`
  - puis les éléments `<b>`
  - puis tous les autres

Par exemple:

```

<doc>
  <b>cons</b>
  <c>tion</c>
  <a>Anti</a>
  <b>titu</b>
  <e>nelle</e>
  <d>ment</d>
</doc>

```

---[XSLT]---

Anticonstitutionnellement

avec l'exemple ci-dessus.

[Corrigé](#)

- À l'aide de `<xsl:element>`, écrire un XSLT qui retranscrit un document XML (la sortie est donc un document XML); on garde les enfants directs de la racine d'un document XML; avec leurs contenus texte, mais sans leurs attributs ni leurs éléments.

Par exemple:

```

<racine coucou="1,2,1,2">
  <X hop="la" bonjour="1">Bla
    <Y hop="li">blabla</Y>
  </X>
  <Z>et encore bla</Z>
</racine>

```

---[XSLT]---

```

<?xml version="1.0"?>
<X>Bla
  </X><Z>et encore bla</Z>

```

avec l'exemple ci-dessus.

[Corrigé](#)

- À l'aide des `<xsl:`

C

m

tf

(avec leurs valeurs originelles).

Par exemple:

```
<racine coucou="1,2,1,2">
  <X hop="la" bonjour="1">Bla
    <Y hop="li">blabla</Y>
  </X>
  <Z>et encore bla</Z>
</racine>
```

---[XSLT]--->

```
<?xml version="1.0"?>
<X hop="la" bonjour="1">Bla
  </X><Z>et encore bla</Z>
```

avec l'exemple ci-dessus.

[Corrigé](#)

4. (\*) En utilisant de la récursion, modifiez le XSLT précédent pour que le XML source soit entièrement recopié; i.e. toute la hierarchie des éléments, depuis la racine. Le point délicat sera d'éviter la redondance du contenu texte; on pourra pour cela utiliser une règle qui `match` les noeuds texte. Par exemple:

```
<racine coucou="1,2,1,2">
  <X hop="la" bonjour="1">Bla
    <Y hop="li">blabla</Y>
  </X>
  <Z>et encore bla</Z>
</racine>
--
[ XSLT ]- <?xml version="1.0"?>
-->      <racine coucou="1,2,1,2"><X hop="la" bonjour="1">Bla
          <Y hop="li">blabla</Y></X><Z>et encore bla</Z></racine>
```

avec l'exemple ci-dessus.

[Corrigé](#)